# Data Handling with Housing Data

September 29, 2025

[1]:
```
import pandas as pd
import numpy as np
```

[3]:
```
df = pd.read_csv('Housing.csv')
```

[5]:
```
df = df.select_dtypes(include=np.number) # Keeps only numeric columns
df.head() # Display the first few rows of the resulting DataFrame
```

[5]:

| | LotArea | MSSubClass |
|---|---|---|
| 0 | 8450 | 60 |
| 1 | 9600 | 20 |
| 2 | 11250 | 60 |
| 3 | 9550 | 70 |
| 4 | 14260 | 60 |

[6]:
```
from sklearn.preprocessing import MinMaxScaler # To scale numeric features to a␣
↪fixed range (0 to 1)
```

[7]:
```
scaler = MinMaxScaler() # Creating a MinMaxScaler object
scaled_data = scaler.fit_transform(df) # Fit the scaler to df and transform␣
↪values to range 0-1
scaled_df = pd.DataFrame(scaled_data, columns=df.columns) # Convert the scaled␣
↪array back to a DataFrame with original column names

scaled_df.head() # Display the first 5 rows of the scaled DataFrame
```

[7]:

| | LotArea | MSSubClass |
|---|---|---|
| 0 | 0.033420 | 0.235294 |
| 1 | 0.038795 | 0.000000 |
| 2 | 0.046507 | 0.235294 |
| 3 | 0.038561 | 0.294118 |
| 4 | 0.060576 | 0.235294 |

[8]:
```
from sklearn.preprocessing import StandardScaler # Import StandardScaler to␣
↪standardize features (mean=0, std=1) for better ML performance
```

[9]:
```
scaler = StandardScaler() # Create a StandardScaler object to standardize␣
↪features
```

```
scaled_data = scaler.fit_transform(df) # Fit the scaler to the DataFrame and␣
 ↪transform the data
# Each column will have mean=0 and standard deviation=1
scaled_df = pd.DataFrame(scaled_data, columns=df.columns) # Convert the scaled␣
 ↪array back to a DataFrame with the original column names

print(scaled_df.head()) # Display the first 5 rows of the standardized DataFrame
```

```
    LotArea  MSSubClass
0 -0.207142    0.073375
1 -0.091886   -0.872563
2  0.073480    0.073375
3 -0.096897    0.309859
4  0.375148    0.073375
```

[11]:
```
# Summary statistics for numeric columns
print(df.describe())
```

```
             LotArea   MSSubClass
count    1460.000000  1460.000000
mean    10516.828082    56.897260
std      9981.264932    42.300571
min      1300.000000    20.000000
25%      7553.500000    20.000000
50%      9478.500000    50.000000
75%     11601.500000    70.000000
max    215245.000000   190.000000
```

[12]:
```
# Get all unique values from a column in a DataFrame
# Syntax:
# dataframe['column_name'].unique()
# Example: Get unique values from the 'LotArea' column
unique_lotarea = df['LotArea'].unique()
# Print the unique values
print(unique_lotarea)  # Outputs a NumPy array containing all distinct LotArea␣
 ↪values
```

```
[ 8450  9600 11250 … 17217 13175  9717]
```

[13]:
```
# Count the number of unique values in a column of a DataFrame
# Syntax:
# dataframe_name['column_name'].nunique()
# Example: Count unique values in the 'LotArea' column
num_unique_lotarea = df['LotArea'].nunique()
# Print the number of unique values
print(num_unique_lotarea)  # Outputs an integer representing how many distinct␣
 ↪LotArea values exist
```

```
1073
```

[14]:
```
# Example: Count unique values in the 'MSSubClass' column
num_unique_mssubclass = df['MSSubClass'].nunique()

# Print the number of unique values
print(num_unique_mssubclass)  # Outputs an integer representing how many␣
 ↪distinct MSSubClass values exist
```

```
15
```

[15]:
```
# Get a concise summary of the DataFrame
# Syntax:
# dataframe.info()
# Example: Get info about the DataFrame 'df'
df.info()
# What it does:
# - Displays the number of rows and columns
# - Lists all column names
# - Shows data types of each column
# - Shows the number of non-null values per column
# - Provides memory usage of the DataFrame
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   LotArea     1460 non-null   int64
 1   MSSubClass  1460 non-null   int64
dtypes: int64(2)
memory usage: 22.9 KB
```

[16]:
```
# Get all column names of the DataFrame
# Syntax:
# dataframe.columns
# Example: Display column names of the DataFrame 'df'
print(df.columns)
# What it does:
# - Returns an Index object containing all column names
# - Useful to quickly see which columns are present in the DataFrame
```

```
Index(['LotArea', 'MSSubClass'], dtype='object')
```

[17]:
```
# Calculate the sum of all values in a specific column of the DataFrame
# Syntax:
# dataframe['column_name'].sum()
# Example: Sum all values in the 'LotArea' column
total_lotarea = df['LotArea'].sum()
# Print the total sum
print(total_lotarea)  # Outputs the sum of all LotArea values in the DataFrame
```

15354569

[18]:
```
# Example: Sum all values in the 'MSSubClass' column
total_mssubclass = df['MSSubClass'].sum()

# Print the total sum
print(total_mssubclass)  # Outputs the sum of all MSSubClass values in the␣
 ↪DataFrame
```

83070

[19]:
```
# Calculate the mean (average) of all values in a specific column of the␣
 ↪DataFrame
# Syntax:
# dataframe['column_name'].mean()
# Example: Find the mean of the 'LotArea' column
mean_lotarea = df['LotArea'].mean()
# Print the mean value
print(mean_lotarea)  # Outputs the average value of LotArea in the DataFrame
```

10516.828082191782

[20]:
```
# Example: Find the mean of the 'MSSubClass' column
mean_mssubclass = df['MSSubClass'].mean()

# Print the mean value
print(mean_mssubclass)  # Outputs the average value of MSSubClass in the␣
 ↪DataFrame
```

56.897260273972606

[21]:
```
# Calculate the standard deviation of all values in a specific column of the␣
 ↪DataFrame

# Syntax:
# dataframe['column_name'].std()

# Example: Find the standard deviation of the 'LotArea' column
std_lotarea = df['LotArea'].std()

# Print the standard deviation
print(std_lotarea)  # Outputs the standard deviation of LotArea values in the␣
 ↪DataFrame
```

9981.264932379147

[22]:
```
# Example: Find the standard deviation of the 'MSSubClass' column
std_mssubclass = df['MSSubClass'].std()
```

```
# Print the standard deviation
print(std_mssubclass)  # Outputs the standard deviation of MSSubClass values in␣
 ↪the DataFrame
```

42.30057099381035

[24]:
```
# Calculate the variance of all values in a specific column of the DataFrame

# Syntax:
# dataframe['column_name'].var()

# Example: Find the variance of the 'LotArea' column
var_lotarea = df['LotArea'].var()

# Print the variance
print(var_lotarea)  # Outputs the variance of LotArea values in the DataFrame
```

99625649.6503417

[25]:
```
# Calculate the variance of all values in the second column 'MSSubClass'
# Syntax:
# dataframe['column_name'].var()
# Example: Find the variance of the 'MSSubClass' column
var_mssubclass = df['MSSubClass'].var()
# Print the variance
print(var_mssubclass)  # Outputs the variance of MSSubClass values in the␣
 ↪DataFrame
```

1789.338306402389

[26]:
```
# Calculate the minimum value of all values in a column
# Syntax:
# dataframe['column_name'].min()
# Example: Find the minimum value of the 'LotArea' column
min_lotarea = df['LotArea'].min()
# Print the minimum value
print(min_lotarea)  # Outputs the smallest value in LotArea
```

1300

[27]:
```
# Calculate the minimum value of the 2nd column (MSSubClass)
min_mssubclass = df['MSSubClass'].min()
# Print the minimum value
print(min_mssubclass)  # Outputs the smallest value in MSSubClass
```

20

[28]:
```
# Syntax: dataframe['column_name'].max()
# Here: df['LotArea'].max() → finds the maximum value in the 'LotArea' column
```

```
# Find the maximum value in the first column 'LotArea'
max_lotarea = df['LotArea'].max()

# Print the result
print("Maximum value in LotArea column:", max_lotarea)
```

```
Maximum value in LotArea column: 215245
```

[29]:
```
# Syntax: dataframe['column_name'].max()
# Here: df['MSSubClass'].max() → finds the maximum value in the 'MSSubClass'␣
 ↪column

# Find the maximum value in the second column 'MSSubClass'
max_mssubclass = df['MSSubClass'].max()

# Print the result
print("Maximum value in MSSubClass column:", max_mssubclass)
```

```
Maximum value in MSSubClass column: 190
```

[30]:
```
# 1. Get minimum value of each column
print("Minimum value of each column:")
print(df.min())
print("----------------------------------------")

# 2. Get minimum value of a particular column (LotArea)
print("Minimum value of LotArea column:")
print(df['LotArea'].min())
print("----------------------------------------")

# 3. Get maximum values of each column
print("Maximum value of each column:")
print(df.max())
print("----------------------------------------")

# 4. Get the sum of each column
print("Sum of each column:")
print(df.sum())
print("----------------------------------------")

# 5. Get the count of non-null values in each column
print("Count of non-null values in each column:")
print(df.count())
print("----------------------------------------")

# 6. Get standard deviation of each numeric column
print("Standard deviation of each numeric column:")
print(df.std())
```

```
print("----------------------------------------")

# 7. Get variance of each numeric column
print("Variance of each numeric column:")
print(df.var())
```

```
Minimum value of each column:
LotArea       1300
MSSubClass      20
dtype: int64
----------------------------------------
Minimum value of LotArea column:
1300
----------------------------------------
Maximum value of each column:
LotArea       215245
MSSubClass       190
dtype: int64
----------------------------------------
Sum of each column:
LotArea       15354569
MSSubClass       83070
dtype: int64
----------------------------------------
Count of non-null values in each column:
LotArea       1460
MSSubClass    1460
dtype: int64
----------------------------------------
Standard deviation of each numeric column:
LotArea       9981.264932
MSSubClass      42.300571
dtype: float64
----------------------------------------
Variance of each numeric column:
LotArea       9.962565e+07
MSSubClass    1.789338e+03
dtype: float64
```

[31]:
```
# Group by MSSubClass and get the first LotArea in each group
print("First LotArea in each MSSubClass group:")
print(df.groupby('MSSubClass')['LotArea'].first())

print("---------------------------------")

# Sum of LotArea for each MSSubClass
print("Sum of LotArea per MSSubClass:")
```

```python
print(df.groupby('MSSubClass')['LotArea'].sum())

print("---------------------------------")

# Count of rows per MSSubClass
print("Count of rows per MSSubClass:")
print(df.groupby('MSSubClass')['LotArea'].count())
```

```
First LotArea in each MSSubClass group:
MSSubClass
20       9600
30       6324
40       5400
45       6120
50      14115
60       8450
70       9550
75       7200
80       7134
85       9180
90      10791
120      4224
160      2645
180      1596
190      7420
Name: LotArea, dtype: int64
---------------------------------
Sum of LotArea per MSSubClass:
MSSubClass
20      6300953
30       544306
40        50239
45        83388
50      1508997
60      3605365
70       607210
75       186108
80       631938
85       186349
90       505443
120      468404
160      170058
180       22474
190      483337
Name: LotArea, dtype: int64
---------------------------------
Count of rows per MSSubClass:
MSSubClass
```

```
20     536
30      69
40       4
45      12
50     144
60     299
70      60
75      16
80      58
85      20
90      52
120     87
160     63
180     10
190     30
Name: LotArea, dtype: int64
```

[34]:
```python
# Aggregate multiple statistics for LotArea and MSSubClass by MSSubClass
grouped_stats = df.groupby('MSSubClass').agg({
    'LotArea': ['sum', 'mean', 'min', 'max'],
    'MSSubClass': 'count'  # count how many rows in each group
})

print(grouped_stats)
```

```
            LotArea                              MSSubClass
                sum          mean   min     max      count
MSSubClass
20          6300953  11755.509328  3182  215245        536
30           544306   7888.492754  3636   25339         69
40            50239  12559.750000  5400   23595          4
45            83388   6949.000000  4388   10594         12
50          1508997  10479.145833  3500  159000        144
60          3605365  12058.076923  3378   63887        299
70           607210  10120.166667  2500   24090         60
75           186108  11631.750000  5520   25419         16
80           631938  10895.482759  6930   21453         58
85           186349   9317.450000  7200   16647         20
90           505443   9720.057692  6040   25000         52
120          468404   5383.954023  2887   14963         87
160          170058   2699.333333  1300   10762         63
180           22474   2247.400000  1477    3675         10
190          483337  16111.233333  4456  164660         30
```

[33]:
```python
# Group by 'MSSubClass' and get the minimum LotArea in each group
print("Minimum LotArea for each MSSubClass:")
print(df.groupby('MSSubClass')['LotArea'].min())
print("------------------------")
```

```
# Group by 'LotArea' and get the minimum MSSubClass in each group
# (only meaningful if there are repeated LotArea values)
print("Minimum MSSubClass for each LotArea:")
print(df.groupby('LotArea')['MSSubClass'].min())
print("-------------------------")
```

```
Minimum LotArea for each MSSubClass:
MSSubClass
20     3182
30     3636
40     5400
45     4388
50     3500
60     3378
70     2500
75     5520
80     6930
85     7200
90     6040
120    2887
160    1300
180    1477
190    4456
Name: LotArea, dtype: int64
-------------------------
Minimum MSSubClass for each LotArea:
LotArea
1300      160
1477      180
1491      180
1526      180
1533      160
         …
70761      20
115149     20
159000     50
164660    190
215245     20
Name: MSSubClass, Length: 1073, dtype: int64
-------------------------
```

[ ]: